



# **NAVAL POSTGRADUATE SCHOOL**

**MONTEREY, CALIFORNIA**

**A SURVEY OF XOR AS A DIGITAL OBFUSCATION  
TECHNIQUE IN A CORPUS OF REAL DATA**

by

Carolina Zarate  
Simson L. Garfinkel  
Aubin Heffernan  
Kyle Gorak  
Scott Horras

January 17, 2014

**Approved for public release; distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

**NAVAL POSTGRADUATE SCHOOL**  
**Monterey, California 93943-5000**

Ronald A. Route  
President

Douglas A. Hensler  
Provost

The report entitled "*A Survey of XOR as a Digital Obfuscation Technique in a Corpus of Real Data*" was prepared for and funded by Department of the Navy.

**Further distribution of all or part of this report is authorized.**

**This report was prepared by:**

Caroline Zarate

Simson L. Garfinkel

Aubin Heffernan

Kyle Gorak

Scott Horras

**Reviewed by:**

**Released by:**

Peter Denning, Chairman  
Computer Science

Jeffrey D. Paduan  
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 17-1-2014			2. REPORT TYPE Technical Report		3. DATES COVERED (From — To) 2013-05-01—2013-09-01	
4. TITLE AND SUBTITLE  A survey of XOR as a digital obfuscation technique in a corpus of real data					5a. CONTRACT NUMBER	
					5b. GRANT NUMBER	
					5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Carolina Zarate, Simson L. Garfinkel, Aubin Heffernan, Kyle Gorak, Scott Horras					5d. PROJECT NUMBER	
					5e. TASK NUMBER	
					5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Naval Postgraduate School Monterey, CA 93943					8. PERFORMING ORGANIZATION REPORT NUMBER  NPS-CS-13-005	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)  The Department of the Navy					10. SPONSOR/MONITOR'S ACRONYM(S)	
					11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT  Approved for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES  The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.						
14. ABSTRACT  To determine the usage of XOR and the need to adapt additional tools, we analyzed 2,411 drive images from devices acquired around the world for the use of bitwise XOR as an obfuscation technique. Using a modified version of the open source digital forensics tool bulk extractor, evidence of XOR obfuscation was found on 698 drive images, with a maximum of 21,031 XOR-obfuscated features on a single drive. XOR usage in our corpus was observed in files with timestamps between the years 1995 and 2009, but the majority use was found in unallocated space. On the corpus tested, XOR obfuscation was used to circumvent malware detection and reverse engineering, to hide information that was apparently being exfiltrated, and by malware detection tools for their quarantine directory and to distribute malware signatures. We conclude that XOR obfuscation is important to consider when performing malware investigations.						
15. SUBJECT TERMS  XOR						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES  28	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code)	

THIS PAGE INTENTIONALLY LEFT BLANK

# Table of Contents

<b>1</b>	<b>Introduction . . . . .</b>	<b>1</b>
<b>2</b>	<b>Prior Work . . . . .</b>	<b>2</b>
<b>3</b>	<b>Materials and Methods . . . . .</b>	<b>5</b>
<b>4</b>	<b>Findings and Discussion . . . . .</b>	<b>8</b>
<b>5</b>	<b>Conclusion . . . . .</b>	<b>12</b>
	List of References . . . . .	17

THIS PAGE INTENTIONALLY LEFT BLANK

## Abstract

To determine the usage of XOR and the need to adapt additional tools, we analyzed 2,411 drive images from devices acquired around the world for the use of bitwise XOR as an obfuscation technique. Using a modified version of the open source digital forensics tool *bulk\_extractor*, evidence of XOR obfuscation was found on 698 drive images, with a maximum of 21,031 XOR-obfuscated features on a single drive. XOR usage in our corpus was observed in files with timestamps between the years 1995 and 2009, but the majority use was found in unallocated space. On the corpus tested, XOR obfuscation was used to circumvent malware detection and reverse engineering, to hide information that was apparently being exfiltrated, and by malware detection tools for their quarantine directory and to distribute malware signatures. We conclude that XOR obfuscation is important to consider when performing malware investigations.

## 1 Introduction

There exist a variety of single-byte operators that have traditionally been used for simple data hiding or obfuscation. One classic technique is the “ROT13” operator [26], which was used to hide jokes transmitted on the early Internet. More common techniques today include single-byte XOR and single-byte rotate operators. These operators can be thought of as poor encryption algorithms with 8-bit keys: they are trivial to decode, but to do so the analyst or tool must specifically probe for evidence of their use. If no detection algorithm is employed, even 8-bit encryption is sufficient to hide data.

The only digital forensics tools known to the authors that provide forensic investigators with an automated approach for finding XOR-obfuscated data are *DCCI\_Carver* and *DC3\_Carver*, two general purpose forensic carving tools developed by the Defense Cyber Crime Institute, the research and development arm of the Defense Cyber Crime Center (DC3).

Current commercial and open source digital forensics tools largely ignore these single-byte obfuscation techniques. Although it is relatively easy to modify forensic and anti-malware tools to scan for data that have been ofuscated, most digital forensic tools and malware scanners eschew steganography detection because it is computationally expensive and the incidence of such techniques is thought to be low. However, simple obfuscation techniques are sufficient to bypass most commercial and open source malware detection techniques, making the lack of de-obfuscation features in tools a vulnerability.

This study focuses on a specific obfuscation technique that we call XOR(255). XOR is the exclusive-or binary operation. XOR binary operations are performed with “keys.” A *key* is a series of bytes with which a source file is XORed with. XOR(255) is a special case of a single-byte key where the key is the hex byte value 0xFF. Bitwise XOR(255) has the effect of inverting every bit in the source file. XOR(255) has the advantage of being fast (it typically

executes in less than 1 clock cycle on modern architectures), reversible, and can be performed in-place. XOR(255) has the additional property of leaving a file’s entropy unchanged, allowing processed data to remain invisible to tools that search for encrypted data using entropy techniques. We focused on XOR(255) because of its ease of use and effectiveness at obscuring information.

For this study we created a plug-in for the open source tool *bulk\_extractor* [16] that processes all data with XOR(255). The structure of *bulk\_extractor* is such that each scanner is applied independently to each block of data during processing. We used the modified tool to scan a corpus of several thousand drive images from used storage devices that had been purchased on the secondary market. We limited our analysis to email addresses, URLs, JPEG digital photographs, Windows executables, ZIP files, and other kinds of easily recognizable information that had been obfuscated by XOR(255). Our tool optimistically applied XOR(255) both before and after every decompression or decoding step in its search for recognizable structured data.

We found that the use of XOR(255) was limited to a small fraction of the corpus, however the uses that we found could be significant in a variety of forensic investigations. We discovered obfuscation used to hide signatures in malware in addition to obfuscated user-generated content that appeared to be targeted for exfiltration. We also identified cases where obfuscation was used by legitimate programs to prevent reverse-engineering. Simple obfuscation techniques were present on drives purchased on three continents over a ten-year period, with some examples as recent as 2009. By its nature, our sampling technique does not provide us with significant amounts of recent data, so we do not know the extent to which this technique is currently in use.

This concludes the introduction. Section 2 is a literature search that discusses uses of XOR obfuscation reported in malware and existing analysis tools. Section 3 provides details regarding our modifications to *bulk\_extractor* and our experimental technique for identifying examples of XOR obfuscation. Section 4 presents our data and our resulting analysis of obfuscated URLs, email addresses, Windows executables, and other data that we found. Section 5 concludes with our recommendations and opportunities for future work.

## 2 Prior Work

Simple data obfuscation techniques predate the modern Internet: ROT13 was used in the 1980s to distribute the answers to riddles and off-color jokes on the Usenet. Although in recent times the existence of ubiquitous encryption would seem to limit the need and use of simple obfuscation, this does not appear to be the case.

### 2.1 Use of XOR

There are relatively few reports in the academic literature regarding XOR obfuscation. Instead, documented examples of XOR use are largely confined to online blogs of malware investigators. Mueller notes that Norton Antivirus uses XOR to obfuscate its log files and quarantine folder [28]; we verify his observation. Hussey suggests that XOR is sometimes used as an obfuscation

technique to hide data that is being exfiltrated [22]; we found evidence of XOR used for this purpose.

Several types of malware have been reported as using simple obfuscation techniques in order to hide the data they are exfiltrating from a victim host machine. *Trojan.NTESSESS* XORs the result of its commands, including uploaded files, with a changing nine byte key [30]. Even advanced malware, such as *Stuxnet*, *Duqu*, *Flame*, and *Red October*, were observed to use XOR as the basis of a simple obfuscation algorithm to hide data that they were stealing [37]. *Stuxnet* uses a 31-byte key with XOR [15]. *Duqu* XORs data from its keylogger and sends it back to its server [37]. Similar to *Duqu*, *Flame* employs XOR obfuscation techniques on captured data, but contains extensive data-capturing features, such as keystrokes, screenshots, email messages, and conversations recorded through the computer's microphone [37]. *Red October* also used XOR obfuscation techniques to exfiltrate information on Nokia phones and iPhones, networks, and deleted files [37].

Simple obfuscation techniques have also been proven to be popular as a method of protecting malicious code or exfiltrated data from both anti-virus software and investigators. The *Sym-bOS/OpFake.A!tr.dial* malware from Opera Updater is one such example. Apvrille found the Opera Updater malware to contain a 91-byte XOR key [3] as a more complicated algorithm for hiding itself. Similarly, variations of both *Trojan.PingBed* and *Trojan.NTESSESS* employ XOR obfuscation and embedding techniques in PNG files as a way of evading anti-virus scanners [29, 30]. Cannell suggests that malware may contain XOR'ed URLs as references to other malicious files and executables online, thus creating the appearance of a benign file [7]. An example is *Trojan.Win32.Patched.mc* that is, which disguises itself as a harmless Flash file. The trojan file analyzed by StopMalvertising, *main.swf*, contained an XOR'ed hexadecimal string, allowing an XOR'ed malicious executable to be downloaded [24]. This poses a problem to anti-virus scanners, as the files appear harmless, but can then download files from URLs known to be malicious. XOR has even been reported to have been used in advanced malware as a method of hiding the main code from anti-virus software. *Stuxnet*, *Duqu*, *Flame*, and *Red October* all obfuscate their payloads and main executables with XOR [37]. In addition, the Storm botnet and other advanced botnets have begun to take advantage of the technique, obfuscating their traffic with XOR in order to avoid identification [23].

Previous research has been conducted on how well anti-virus systems deal with simple obfuscation in malware. Common commercial anti-virus systems tested by Christodorescu and Jha frequently did not identify obfuscated malware, demonstrating that simple obfuscation is an effective technique to avoid anti-virus scanners [9]. An article by the "Internot Security Team" downloaded from the Packet Storm website examined a similar case, but evaluates the effectiveness of common anti-virus scanners with an easily-detectable virus before and after simple obfuscation techniques are employed on the file [34]. The results demonstrated that digital obfuscation techniques allow malware to easily bypass many anti-virus scanners.

Both Cannell and StopMalvertising's findings suggest that there are key strings or patterns, such

as URLs, which can be searched for to help identify and decrypt an XOR obfuscated section.

Others have provided insight into techniques that might help with undermining XOR obfuscation. Cannell suggests elsewhere that patterns inside XOR'ed material can allow one to find the XOR key [6]. Malware normalization has also been proposed by Christodorescu et al. as a method of undermining obfuscation techniques by normalizing the file, allowing anti-virus scanners to be able to detect malicious content [10].

Additionally, the authors recently became aware of research conducted by the Defense Cyber Crime Institute beginning Spring 2010. While investigating an obfuscated malware sample, DCCI researchers extracted two XOR(255) executables and one XOR(255) Microsoft Word document. The obfuscated executables were not surprising because it was known at the time that XOR(255) was a hiding technique that was sometimes used by malware developers, but uncovering an inverted Microsoft Word document was unexpected. In June 2010, DCCI developers included in that organization's general purpose carving tools, *DCCI\_Carver* and *DC3\_Carver*, a feature that allows examiners to optionally XOR the data set being carved with any hex byte value and re-carve the data for all file types of interest. A special feature was also added to allow for the carving of XOR(255) obfuscated files, since this obfuscation technique was considered to be more common than using other hex byte values. During testing, DCCI found XOR(255) obfuscation on approximately one-third of the data sets it uses to test with. The types of inverted files that were uncovered included executables; DLLs; Web pages; general text-based files; source code; JPG, PNG, GIF, and BMP icons; and byte strings that were identified as MPEG video fragments.

Since the Fall of 2010, in the hopes of fostering a general interest among digital researchers and investigators in looking for the source of XOR obfuscated data and the extent to which it is used, DCCI researchers have been reporting their findings at many different venues [21], including forensic conferences, forums, tool expos, technical exchange meetings, and Webcasts, and to students attending the Defense Cyber Investigations Training Academy (DCITA) Data Recovery and Advanced Forensics Concepts classes.

## **2.2 Tools for manual analysis**

Although we found numerous digital forensics tools that can de-obfuscate a region of bytes with an XOR mask or use XOR pre-processing in malware analysis, only one applies XOR as part of automated processing. The remainder of the tools that provide some way of de-obfuscating data require that these techniques be manually invoked by the operator.

Some tools offer methods of de-obfuscating data, but leave the interpretation up to the investigator. Hex editors oftentimes implement such a feature, allowing the user to view data in hex, and offering de-obfuscation functions. The open source Hexplorer [12] has the ability to XOR regions before they are displayed. Didier Steven's *translate.py* [32] script also allows investigators to XOR a file with a single-byte key. Other tools are specifically tailored for de-obfuscating certain components of a forensic investigation. The MemoryDump [2] plugin, for instance, has

an option of XORing a memory dump before outputting the data for the user.

Other tools examine data to look for XOR keys, leaving it up to the user to de-obfuscate and analyze the information. XORBruteForcer [14] uses a brute-force algorithm to determine the keys that were most likely to have been used in XORing the data. NoMoreXOR [13], in contrast, tries to guess an XOR key to the best of its ability using frequency analysis.

String searching is also a frequently used feature in obfuscation analysis tools. Steven’s XOR-Search [31] tool can search using a brute-force approach for specific strings that have been obfuscated using XOR, ROL, or ROR. Similarly, Steven’s other tool, XORStrings [33], is an extension of the XORSearch tool and scans a given file for strings that have been XOR’ed with a specific sequence. The iheartxor [20] offers a similar function, searching with brute-force for certain obfuscated regular expressions and plain strings.

## 2.3 Tools with automated analysis

As mentioned previously (§2.1), the only digital forensic tools known to the authors that provide investigators with an automated approach to uncovering a wide variety of XOR obfuscated files from suspect media are *DCCI\_Carver* and *DC3\_Carver*. In addition to these tools, our literature search turned up tools that provide automated help in identifying obfuscated malicious files. Several programs can search for XOR’ed executables embedded in files. Hexacorn Ltd.’s DeXRAY [25] acts as a file carver and can identify obfuscated malicious files under a single layer of XOR. The OfficeMalScanner [4] has a feature to scan for traces of XOR’ed malicious data within office documents using a brute-force method. Another document scanner tool, Cryptam [35], uses cryptanalysis in order to identify obfuscated embedded executable files, and has been proven effective in detecting XOR obfuscation [36].

# 3 Materials and Methods

We modified an open source tool to preprocess all examined data with XOR(255). We then processed data from 2,411 forensic images and found 324,144 XOR artifacts on 698 of them, with a maximum of 21,031 validated artifacts on a single forensic image.

The goal of the research was to identify the different ways that XOR obfuscation was used in real data in order to determine whether the quantity and quality of the cases of obfuscation were sufficient enough to suggest implementing XOR de-obfuscation functions as a standard step in automated forensic processing. We analyzed the corpus of data, asking to what extent XOR(255) was implemented as an obfuscation technique.

## 3.1 Real Data Corpus

The Real Data Corpus is a collection of several thousand digitized digital storage devices collected from several countries worldwide between 1998 and 2013 [17]. The corpus consists of data from computer hard drives, cell phones, CD-ROMs, DVDs, and thumb drives purchased

from secondhand computer stores and markets. The RDC thus allows us to sample XOR usage over a 15 year period in a variety of countries around the world, essentially affecting a real-world set of data.

A limitation of this corpus is that it does not include data sets that are known to have been used by criminals, terrorists, or other types of subversives, and as such it will be difficult to draw conclusions regarding the importance of extracting and examining XOR obfuscated files in criminal, counterintelligence, and counterterrorism cases without further research. In addition, since we limited our analysis to email addresses, URLs, JPEG digital photographs, Windows executables, ZIP files, and other kinds of easily recognizable information, we did not look for all types of files that would be considered relevant to these types of investigations.

### 3.2 The *bulk\_extractor*

The *bulk\_extractor* is a forensic tool that can scan digital media for information including email addresses, phone numbers, Internet domain names, and other data that would be of use to investigators. The *bulk\_extractor* is employed by both triage and cyber malware investigators [18]. Triage investigators can look for identities on systems and use the data carved by *bulk\_extractor* to find them. Cyber malware investigators look for information that might indicate malicious activity. The tool employs compiled regular expressions and hard-coded FSMs to extract features from digital media. *Features* are pieces of key information such as credit cards, emails, URLs, etc. that might be of importance to forensic investigators. For each feature, *bulk\_extractor* determines the feature's encoding and records that information in its output. The program labels features found using XOR(255) de-obfuscation with the string "XOR(255)".

The *bulk\_extractor* framework tracks which features result from the application of each scanner; this information is recorded in the program's output. By using *bulk\_extractor* in this manner, it was possible to search systematically throughout our entire corpus for different kinds of XOR obfuscation uses. For example, we could detect and distinguish the case of a JPEG photograph being XOR'ed and then archived in a ZIP file from a series of JPEGs archived in a ZIP file and then XOR'ed.

We used the *identify\_filenames.py* tool included in the *bulk\_extractor* package to associate identified features with the files they are in. For each feature, the tool indicated if the feature occurred in an allocated file, a deleted file, or disk sectors that could not be associated with any file. This allowed us to identify the files containing the XOR'ed information and better understand the purpose and extent of the obfuscation.

### 3.3 Feature selection

We examined the encoding of domains, URLs, email addresses, ZIP file data structures, and Windows Portable Executable (WINPE) headers in order to find examples of XOR(255) usage.

The problem with looking for features that have been XOR'ed is being able to distinguish actual XOR'ed features from "false positives," or data that appear by chance in a form the scanner is searching for, yet are not actual features.

URLs
WINPE headers
The local file headers of ZIP archives
Exif headers (from JPEGs)

Table 1: *bulk\_extractor* features used in this study. Each of these features have a high degree of internal structure and are thus self-validating. As a result, they are extracted with an extremely low false positive error rate.

We observed a much greater percentage of false positives (that is, significantly lower *precision*) among the features that were found with XOR encoding than is typical for most *bulk\_extractor* output. The *bulk\_extractor* output normally has a high precision. We assume that the error rate, or the amount of false positives detected by *bulk\_extractor*, is constant for random input and a large enough set of data. When the input contains a substantial signal, such as with running *bulk\_extractor* to find plaintext features, the precision is high. However, with the XOR filter, the signal is low, as the majority of features on the media are not XOR’ed. Since the signal is lower, but the error rate remains constant, the precision is lower. The higher incidence of false positives complicated the analysis. We addressed the error incidence by restricting our analysis to features with significant internal that are automatically checked for internal consistency (table 1). These features are thus self-validating. Self-validating features can also be easily confirmed to be legitimate at a glance, as with opening a JPEG or visiting a site by URL, for instance. Thus, we assume most of these identified features are true positives.

As our goal was to evaluate files containing certain XOR(255)-encoded features, we ignored types of features that we were not evaluating in order to restrict our data set. We removed features that were not XOR’ed, types of features that we were not looking at, and “false positive” features that we did not recognize as legitimate features.

### 3.4 Analysis of identified files

Once we identified features that had been XOR-encoded, we proceeded to identify the files in which they were contained. The Sleuth Kit [8] allowed for us to extract files that *identify\_filenames.py* associated with intriguing XOR’ed features. We analyzed the original files and the files after decoding with XOR(255) to make a conclusion about the intention with which the XOR obfuscation was used.

### 3.5 Evaluation of URLs

The vast majority of the XOR-encoded features that we found were URLs. Malware files may contain some mechanisms to further infiltrate a victim, while appearing benign at the same time. For example, malware frequently contained URLs of malicious sites to download additional malicious executables or other files. We examined the XOR’ed URL features outputted by *bulk\_extractor*, as it has been reported that some malware samples use XOR to obscure embedded URLs. We evaluated the XOR’ed URLs using Google Safe Browsing API [19] and the McAfee TrustedSource Real-Time threat service [27].

## 4 Findings and Discussion

### 4.1 XOR usage by time and by country

Although the majority of XORed features were found in unallocated space, approximately 10% were found in files that were either allocated or whose filename and file timestamps could be recovered using The SleuthKit. Table 2 shows the distribution of XOR features by year, where the year corresponds to the year of the modification time of the file in which the XOR'ed feature was found. As can be seen from the table, modification timestamps are occasionally invalid. We believe that files with modification timestamps before the year 1996 or after 2013 almost certainly are the result of improperly set clocks or data corruption. Nevertheless, the majority of the timestamps appear reasonable and indicate significant use of XOR as an obfuscation technique between 2004 and 2009. These dates should be viewed as a qualitative result, not quantitative, as the incidence of XORed features by year has not been normalized with respect to the representation of these years within the Real Data Corpus. (Such normalization, while possible, would not be meaningful given the manner in which the RDC was acquired.)

Table 3 shows the distribution of XOR features by country in which the forensic images was acquired. As before, these are absolute counts and are not normalized by the representation of each image within the Real Data Corpus.

### 4.2 XOR obfuscation as a watermark

We found several cases in which an innocuous URL was XOR(255)-encoded and embedded in a legitimate program. For example, drive image *AE10-1023* contained a copy of Nero 7 with contained byte sequence `97 8b 8b 8f c5 d0 d0 88 88 88 d1 91 9a 8b 90 d1 9c 90 92 0a` at decimal offset 15,416,790,675. This maps to byte offset 112 of the file `Program Files/Nero/Nero 7/Nero CoverDesigner/def.dat`, a 13630 byte file with the MD5 of `8f4b534ed6a82e1885e155541aab0940` that is reported to be part of the Nero Premium distribution [1]. Transformed with XOR(255) the string becomes `http://www.nero.com`.

### 4.3 XOR'ed URL analysis

In total, there were 281,712 XOR'ed URLs found in the drive images, of which 30,684 were distinct.

Google's Safe Browsing (GSB) database is only checks for the domains of sites and is quite conservative. For each XOR'ed URL, GSB only reports whether the URL's domain is "ok" or "malware." McAfee's service considers the entire URL and provides better discrimination. For each URL we recorded whether McAfee considered it to be "High Risk," "Medium Risk," "Minimal Risk" or "Unverified."

Table 4 provides an analysis of how the URLs were evaluated by the two services. Overall, roughly 10% of the distinct URLs found in XORed data were malware.

## 4.4 XOR obfuscation in anti-virus software

Confirming Mueller’s blog posting, we found many XOR-obfuscated URLs in the Norton Anti-Virus log files and virus definitions. We also found files in the “Quarantine” directory had been obfuscated with XOR(255). RDC drive image *il3-0161* contained one such collection of XORed quarantined malware. The *bulk\_extractor* determined that the file *05FB1F54.exe* to contain an XORed WINPE header. The file was also indicated to be located under `Program Files/Norton Antivirus/Quarantine/` with the MD5 hash value *a96ae9519ea9-68ac0089d6b53cef9b2b*. We performed a hex dump on the original file, and it appeared to have no executable code or strings. However, once we XOR’ed the file, the entire file appeared to be malware, containing strings such as “RegDeleteKeyA”, “DownloadFile”, and “Download.Trojan”.

One of the primary vectors through which malware is distributed are “drive-by downloads” from compromised web servers, and thus a key indicator of malware is the presence of a malicious URL. Norton and other anti-virus scanners use this as a method of identifying malware. By obfuscating malware signatures and log files, anti-virus scanners avoid accidentally identifying themselves as malware.

## 4.5 Obfuscated URLs in malware

We found a significant number of XOR’ed domains and URLs within programs that were clearly associated with malware. In many cases the URLs were either verified as malicious by Google Safe Browsing API or had names that were clearly malicious. One such example is in RDC drive image *IN10-0145* with the file *SENDFILE.EXE*, which has a MD5 of *3d419f96355b93e6-41ba097c08121937*. The unobfuscated version of *SENDFILE.EXE* contained several malicious strings, including URLs linked to malware and several instances of modifying the registry and processes. For instance, the 47 byte sequence `b7 ab ab af c5 d0 d0 ac bc be b1 b1 ba ad d1 a9 d2 a7 d2 ac bc be b1 b1 ba ad d1 bc b0 b2 d0 ac ba ab aa af d0 b9 b6 b3 ba d1 af b7 af` appears at the decimal offset 4426 in *SENDFILE.EXE*. Processed with XOR(255), this string becomes `HTTP://SCANNER.VAV-X-SCANNER.COM/SETUP/FILE.PHP`. Based on this analysis, it is clear that some malware are using XOR(255) obfuscation in their normal operations.

The use of XOR(255) by both anti-virus scanners and malware is problematic. In our testing, many program samples containing malicious XOR’ed URLs were not identified as malicious by *any* of the 46 anti-virus engines at VirusTotal.com. However, after we de-obfuscated the files, many of the engines would then identify the malware. Clearly, the engines are simply looking for the malicious URLs (since applying XOR(255) to the entire executable would also corrupt the Windows PE header and damage the program’s executable code.) It is troubling that common anti-virus engines and professional forensic tools do not even offer this simple technique for finding obfuscated malicious data, especially given the widespread use of XOR(255) in malware that we observed in the Real Data Corpus and in our literature search.

XOR’ed domains and URLs suggested malware at work. Series of repeating domains appeared

in several of the RDC drive images. Simple searches of the domains and running them through VirusTotal indicated that they were malicious sites. We identified the files containing the patterns of the domains and using the `translate.py` tool, we performed a XOR(255) binary operation on the file. The XOR'ed file contained evidence suggesting that it was malware. Embedded PDF, emedded JavaScript, heap oversprays, and invisible IFRAMES with a list of malicious URLs, including URLs from which the repeating domains were extracted from, all appeared XOR'ed in the file. From the evidence in the file, it appears that it is a variation of the Brontok worm.

#### 4.6 XOR obfuscation as anti-reverse engineering

We noted XOR-obfuscated variations of an email address belonging to Dr. Yuriy Reznik, the co-developer of the RealAudio and RealVideo algorithms [11], in DLLs associated with software from his company throughout many of the drive images. The email addresses and its variations would appear repeated several times in blocks throughout the Real codecs. One such example appears in the drive image AE10-1029 in the file `Program Files/Real/RealPlayer/converter/Codecs/erv4.dll`. The `erv4.dll` file is 483328 bytes and has an MD5 of `e8b759859b53e19c261162783dae9869`. The byte sequence `a6 8a 8d 96 86 df ad 9a 85 91 96 94 df c3 86 8d 9a 85 91 96 94 bf 8d 9a 9e 93 d1 9c 90 92 c1 ff` appeared at offset 61120 and was de-obfuscated to name and email address *Yuriy Reznik* <`yreznik@real.com`>. We contacted Dr. Reznik and learned that his obfuscated email address was embedded in the binary as a decryption key that was used for decryption of code tables used by the RealVideo 8 codec.

We also found several instances of XOR(255)-encoded JPEG digital photographs that appeared to be from Adobe Flash video games; RDC drive image *IN10-0060* contained the Battle Rush Adobe Flash game menu screen background, image *SG1-1062* had an XORed menu screen of Gutterball 2, and many Beach Party Craze backgrounds appeared in image *TH0001-0010* drive. We believe that the encoded JPEG pictures were also an attempt at anti-reverse engineering.

#### 4.7 Sensitive Data in a XOR(255)-encoded ZIP file

On a single RDC drive image we observed remnants of multiple ZIP files in unallocated space containing confidential information that had been archived and XOR-encoded. The disk was from a computer running Windows 95 that had been purchased on the secondary market and imaged in 2007. The most recent use of the drive was in 2006.

On the drive image we found a total of 802 documents, with timestamps in the ZIP archive ranging from 1991 through 1999, with the majority of the files from 1998 (112 files) and 1999 (623 files). No archived files were found with timestamps after 1999.

After finding these files, we modified *bulk\_extractor*'s ZIP-scanner to carve the remnants into files that could be transferred to another system and analyzed. Reviewing these extracted files we found spreadsheets, personal emails, and other sensitive documents containing employee names, home addresses, and government identification numbers. We located a batch file,

EMPACA2.BAT, which zipped together the documents to create the archive on the drive image. We could not find evidence of the tools that had been used to obfuscate the archive, and thus could not prove that the obfuscation was the result of an exfiltration attempt.

There are several different possible explanations as to the sensitive data being obfuscated. The data could have belonged to the person that was obfuscating the data, where the owner could have either been developing an app to protect a set of data or to protect their data. The obfuscated ZIP file could be the result of software processing data, as well. On the other hand, this data could have been obfuscated to be hidden by a rogue employee or as an attempt at exfiltrating the sensitive data.

## 4.8 Performance Impact

We saw a significant increase in processing time from the addition of the XOR scanner. One standard test drive image resulted in an additional 53% processing time, while another resulted in an increase of 70% (Table 5). Timing was performed on a 12-core HP Z800 with dual Intel Xenon E5645 CPUs running at 2.4GHz with 24GiB of RAM. The OS was Fedora 19 Linux.

Because *bulk\_extractor* recursively processes decompressed data and the XOR scanner attempts to de-obfuscate at each step of the pipeline, the actual increase in time will be data-dependent for any given disk image. The increase in time is also proportional to the amount of data that is compressed and must be re-processed, not to the amount of data that is XOR-obfuscated. For example, NPS Realistic drive image *nps-2011-2tb* contains a large number of Adobe PDF files, each of which contains multiple *zlib*-compressed regions. Applying XOR deobfuscation to these bytes results in a quadrupling of processing by other *bulk\_extractor* scanners: the bytes are processed with the scanners as they sit in the disk image; they are processed after de-obfuscation with XOR; after decompression; and after both decompression and then de-obfuscation with XOR.

## 4.9 Obfuscated URLs in Norton

In our manual analysis of the drive images containing XOR'ed features, we observed significant usage of XOR in both URL and WINPE header features identified to be located in files associated with Norton or Symantec. Upon further examination, it appeared that Norton was using XOR(255) to encode many of their files.

The Norton logs, WebHist.log and Spam.log, both contain legitimate URL and email features detected by *bulk\_extractor*. Each feature appeared to have been XOR'ed separately, as opposed to the entire file having been encoded.

Norton's virus definitions located in the VirusDefs folder also contained URLs encoded with XOR(255). Many of the URLs found in order in blocks on many of the RDC drive images were from Norton's virus definitions files. One pattern of XOR'ed URLs appeared 206 times across 28 unique drive images from 3 countries. These drive images also contained many other malicious URLs listed in the virus definitions. It is unknown why XOR(255) encoding was

used to obfuscate the virus definitions. Several possibilities as to the purpose of Norton's use of XOR in the virus definitions include protecting the user from accidentally following the URLs, preventing itself from tripping over its own data, and protecting their virus definitions from other anti-virus competitors.

An additional use of XOR encoding by Norton is in quarantined files. The *identify\_filenames.py* script identified several XOR'ed WINPE headers inside of files in Norton's Quarantine folder. Each file was completely XOR'ed and contained strong evidence suggesting it as a malicious file. However, it is unknown how effective quarantining a file with this method would be if the malware itself is using XOR(255) encoding. By XORing the malware, the entire file, if not just parts of the file, would be unXOR'ed.

#### **4.10 Accuracy of the *bulk\_extractor***

We were interested in substantiating the accuracy of *bulk\_extractor*'s ability to detect features obfuscated with XOR(255). Two standard forensic drive images, *nps-2009-domexusers* (40GB) and *nps-2011-2tb* (2TB), were used to study the accuracy of *bulk\_extractor*'s XOR scanner. We took a random sample of 500 features from each type of feature from each drive image. Each feature was determined whether it was a true positive or false positive by hand. We found that for each type of feature, *bulk\_extractor* had either almost all of the features as true positives or all of the features as false positives. There was a higher incidence of features on the 2TB drive image that were almost completely false positives than the 40GB drive image.

#### **4.11 The XOR(255)-UNZIP-XOR(255) Inverse Property**

We were surprised to discover several instances in which the sequence of XOR(255)-UNZIP-XOR(255) applied to a fragment of a ZIP-encoded file resulted in the same result that would be produced by the straightforward application of UNZIP. We hypothesize that zlib's use of an adaptive decompression algorithm is at work here. We subsequently suppressed the XOR(255)-UNZIP-XOR(255) processing with a modification to the *bulk\_extractor* XOR scanner.

## **5 Conclusion**

### **5.1 Summary of Results**

In our analysis of the Real Data Corpus, we have observed multiple cases of XOR(255) obfuscation. XOR(255) was used to encode URLs in both anti-virus scanner files and malware as a method of avoiding detection. XOR(255) obfuscation was also used in deterring others from reverse engineering code in both the RealVideo 8 codecs and several Adobe Flash video games. XOR(255) also was found in a single case that we believe was a successful attempt to exfiltrate sensitive data.

### **5.2 Recommendations**

In our study of the Real Data Corpus for examples of XOR obfuscation, we came across several prominent cases where data from XOR obfuscation may help in an investigation. However,

the increase in processing time by *bulk\_extractor* and other forensic tools may be disadvantageous to investigators examining large sets of data in restricted amounts of time. Because of this, we recommend that although forensic tools should implement features that would allow an investigator to perform simple de-obfuscation on a set of data, the de-obfuscation functions should not be set to run automatically, but should be included as an optional feature (similar to the approach used by DCCI when that organization added the XOR feature to *DCCI\_Carver* and *DC3\_Carver*. However, since our analysis of the data has demonstrated a need for de-obfuscation for both malware and detecting exfiltrated data, we advise that people performing malware investigations or in need of thoroughly analyzing data should always run the forensic tool with the de-obfuscation functions enabled in order to capture all the information. Likewise, those engaged in child exploitation or objectionable content cases should enable XOR deobfuscation to address the so-called “botnet” or “SODDI” defense [5]. Our scanner is included in the *bulk\_extractor* 1.4 release and can be enabled with the flag `-e xor`.

### 5.3 Future Work

We believe that de-obfuscation with *bulk\_extractor*’s XOR scanner can be improved to run with little noticeable performance impact. Some plausible solutions could perform XOR de-obfuscation on only top-level data and not at every level, or only applying the de-obfuscation when certain byte sequences are observed.

Our study of the Real Data Corpus has revealed that XOR is in fact commonly used as an obfuscation technique. It would be useful to perform a survey on the prevalence and purpose of other simple obfuscation and steganography techniques, such as rotate and the ROT13 algorithm, as well as, XOR with different keys.

Lastly, it is necessary that anti-virus systems develop a better way of indicating that something should not be scanned, such as with digital signatures. The evidence of only a simple XOR being sufficient to evade today’s anti-virus system was troubling.

### 5.4 Acknowledgements

We wish to thank Dave Ferguson, the DC3 representative and Deloitte contractor who initially told us about the 2010 DCCI XOR obfuscation research activities. This research effort was undertaken as a direct result of Mr. Ferguson’s December 2012 comments. We also wish to thank Mr. Ferguson and DCCI for their useful comments that helped to improve this paper.

Additionally, we wish to thank COL Greg Conti, LTC Matt Burrow and LTC Michael Nowatkowski at the US Military Academy for supporting the internship program between West Point and the Naval Postgraduate School. We also wish to thank Poolesville High School for the support of its internship program.

Thanks also to Dr. Yuriy Reznik for responding to our email.

Michael Shick at NPS wrote the first prototype of the *bulk\_extractor* XOR plug-in.

This work was performed at the suggestion of the Department of Defense Cyber Crime Center (DC3), which had previously noticed that inverting the data on a disk image and then carving could yield some interesting results.

Robert Beverly at NPS provided useful comments on this document.

The opinions and views expressed in this article are those of the authors and do not reflect the views of the Naval Postgraduate School, the US Military Academy, the Department of the Navy, the Department of the Army, or the US Department of Defense.

Year	# URL	# WinPE	# ZIP	#Exif
1980	4	7	0	0
1981	6	0	0	0
1985	15	0	0	0
1990	0	20	0	0
1996	2	11	0	0
1997	185	15	0	0
1998	443	126	3	0
1999	261	526	44	0
2000	252	526	12	0
2001	593	238	1	0
2002	734	234	1	0
2003	224	87	0	0
2004	1,359	427	34	0
2005	2,640	184	0	0
2006	1,934	3,840	6	0
2007	315	16,782	0	0
2008	1,376	1,973	0	0
2009	1,722	489	0	0
2010	802	468	0	0
2011	14,594	8	74	0
2013	11	1	0	0
2014	49	1	0	0
2016	10	1	0	0
2018	818	0	0	0
2019	3	0	0	0
2023	2	0	0	0
2027	346	0	0	0
2029	4	1	0	0
2030	4	2	0	0
2033	218	0	0	0
2037	14	0	0	0
2080	20	14	0	0
2081	10	2	0	0
no file	252,742	11,550	4,594	130
Total	281,712	37,533	4,769	130

Table 2: Validated XOR features by year for the analyzed forensic images, where the “year” is corresponds to the modification time of the file within which each XOR-encoded feature was found. “no file” indicates that the XOR-encoded features could not be located to a specific file. Timestamps prior to 1996 and after 2011 are likely the result of an improperly set system clock or on-disk corruption and are reported here for completeness.

country	total drives	drives with XOR WinPE	drives with XOR URL	drives with XOR ZIP	drives with XOR exif
BANGLADESH	57	15	5	0	0
BOSNIA AND HERZEGOVINA	7	0	0	0	0
CANADA	48	8	1	0	0
CHINA	807	25	1	0	0
EGYPT	7	2	2	0	0
GERMANY	37	22	6	1	0
GHANA	19	8	1	0	0
GREECE	10	2	0	0	0
INDIA	603	185	77	13	4
ISRAEL	260	84	39	9	0
MEXICO	173	73	16	3	1
MONACO	11	6	2	0	1
NEW ZEALAND	1	0	0	0	0
PAKISTAN	81	31	2	0	0
PALESTINE, STATE OF	140	39	8	3	0
SINGAPORE	34	4	1	0	0
SWITZERLAND	2	0	0	0	0
THAILAND	17	9	1	2	1
TURKEY	10	6	2	0	0
UNITED ARAB EMIRATES	87	62	7	19	0
Total	2,411	581	171	50	7

Table 3: Incidence of forensic images with Validated XOR features, by country

McAfee Risk	Google OK	Google Malware	Total
High Risk	1,938	143	2,081
Medium Risk	1,301	9	1,310
Minimal Risk	24,090	6	24,096
Unverified	2,216	5	2,221
Total	29,545	163	29,708

Table 4: Cross tabulation of how the 30,684 URLs were classified by Google's Safe Browsing API and McAfee's Real-Time Threats database.

test image	Size	without XOR	with XOR	$\Delta$
nps-2009-domexusers	40GB	522 sec	799 sec	+53%
nps-2011-2tb	2TB	34,140 sec	58,147 sec	+70%

Table 5: Observed processing times for *bulk\_extractor* with and without XOR scanner.

## References

- [1] View nero premium details, 2013. <http://www.checkfilename.com/view-details/Nero-Premium/>, Systweak. Last accessed August 23, 2013.
- [2] aeon. MemoryDump, Aug 2009. <http://www.woodmann.com/collaborative/tools/index.php/MemoryDump>. Last accessed August 13, 2013.
- [3] Axelle Apvrille. Symbian malware uses a 91-byte XOR key, Nov 2012. <https://blog.fortinet.com/symbian-malware-uses-a-91-byte-xor-key/>. Last accessed August 12, 2013.
- [4] Frank Boldewin. Frank Boldewin’s [www.reconstructor.org](http://www.reconstructor.org), Sep 2009. <http://www.reconstructor.org/code.html>. Last accessed August 12, 2013.
- [5] Susan W. Brenner, Brian Carrier, and Jef Henninger. The trojan horse defense in cyber-crime cases. *Santa Clara Computer and High Technology Law Journal*, 21:1–53, 2004. <http://digitalcommons.law.scu.edu/chtlj/vol21/iss1/1/>.
- [6] Joshua Cannell. Nowhere to hide: Three methods of XOR obfuscation, 2013. <http://blog.malwarebytes.org/intelligence/2013/05/nowhere-to-hide-three-methods-of-xor-obfuscation/>. Last accessed August 14, 2013.
- [7] Joshua Cannell. Obfuscation: Malware’s best friend, Mar 2013. <http://blog.malwarebytes.org/intelligence/2013/03/obfuscation-malwares-best-friend/>. Last accessed August 13, 2013.
- [8] Brian Carrier. The sleuth kit, 2013. <http://www.sleuthkit.org/sleuthkit/index.php>. Last accessed August 13, 2013.
- [9] Mihai Christodorescu and Somesh Jha. Testing malware detectors. *SIGSOFT Softw. Eng. Notes*, 29(4):34–44, Jul 2004. ISSN 0163-5948. <http://doi.acm.org/10.1145/1013886.1007518>. Last accessed July 2, 2013.
- [10] Mihai Christodorescu, Johannes Kinder, Somesh Jha, Stefan Katzenbeisser, and Helmut Veith. Malware normalization. Technical report, University of Wisconsin and Technische Universität München, Nov 2005. <ftp://ftp.cs.wisc.edu/pub/techreports/2005/TR1539.pdf>. Last accessed August 13, 2013.
- [11] Gregory J Conklin, Gary S Greenbaum, Karl Olav Lillevold, Alan F Lippman, and Yuriy A Reznik. Video coding for streaming media delivery on the internet. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(3):269–281, 2001.
- [12] Marcin Dudek. Hexplorer, 2013. <http://sourceforge.net/projects/hexplorer/>. Last accessed August 13, 2013.

- [13] Glenn Edwards. NoMoreXOR, Jan 2013. <https://github.com/hiddenillusion/NoMoreXOR>. Last accessed August 8, 2013.
- [14] Jose Miguel Esparza. XORBruteForcer, Sep 2008. <http://eternal-todo.com/var/scripts/xorbruteforcer>. Last accessed August 9, 2013.
- [15] Nicolas Falliere, Liam O. Murchu, and Eric Chien. W32.Stuxnet Dossier. Feb 2011. [http://www.symantec.com/content/en/us/enterprise/media/security\\_response/whitepapers/w32\\_stuxnet\\_dossier.pdf](http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf). Last accessed August 9, 2013.
- [16] Simson Garfinkel. Digital media triage with bulk data analysis and bulk\_extractor. *Computers & Security*, 32:57–72, February 2013.
- [17] Simson Garfinkel, Paul Farrell, Vassil Roussev, and George Dinolt. Bringing science to digital forensics with standardized forensic corpora. In *Proceedings of the 9th annual Digital Forensic Research Workshop, DFRWS*, 2009. <http://www.dfrws.org/2009/proceedings/p2-garfinkel.pdf>. Last accessed August 14, 2013.
- [18] Simson L. Garfinkel. Digital media triage with bulk data analysis and bulk\_extractor. *Computers & Security*, 32(0):56 – 72, 2013. ISSN 0167-4048. <http://www.sciencedirect.com/science/article/pii/S0167404812001472>. Last accessed July 2, 2013.
- [19] Google. Safe browsing API, May 2013. <https://developers.google.com/safe-browsing/>.
- [20] Alexander Hanel. iheartxor, 2012. <http://hooked-on-mnemonics.blogspot.com/p/iheartxor.html>. Last accessed August 8, 2013.
- [21] Mark Hirsh. DCCI\_StegCarver. In *Presentation developed for a MITRE-sponsored Technical Exchange Meeting*, September 21, 2010.
- [22] Brian Hussey. Decoding data exfiltration—reversing XOR encryption, 2011. <http://crucialsecurityblog.harris.com/2011/07/06/decoding-data-exfiltration-%E2%80%93-reversing-xor-encryption/>. Last accessed August 14, 2013.
- [23] Brent ByungHoon Kang, Eric Chan-Tin, Christopher P. Lee, James Tyra, Hun Jeong Kang, Chris Nunnery, Zachariah Wadler, Greg Sinclair, Nicholas Hopper, David Dagon, and Yongdae Kim. Towards complete node enumeration in a peer-to-peer botnet. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 23–34. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-394-5. <http://doi.acm.org/10.1145/1533057.1533064>. Last accessed August 12, 2013.
- [24] Kimberly. Analysis of imm32.dll - Trojan.Win32.Patched.mc, Jul 2011. <http://stopmalvertising.com/malware-reports/analysis-of-imm32.dll-trojan.win32.patched.mc.html>. Last accessed August 13, 2013.

- [25] Hexacorn Ltd. DeXRAY, 2012. <http://www.hexacorn.com/blog/category/software-releases/dexray/>. Last accessed August 8, 2013.
- [26] I. Venkata Sai Manoj. Cryptography and steganography, 2010. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.184.5413>. Last accessed August 14, 2013.
- [27] McAfee. Trustedsource—check single url, 2011. <https://www.trustedsource.org/en/feedback/url?action=checksingle>. Last Accessed September 5, 2013.
- [28] Lance Mueller. Xor entire file or selected text, March 17 2008. <http://www.forensickb.com/2008/03/xor-entire-file-or-selected-text.html>.
- [29] Cyber Engineering Services. Malware obfuscated within PNG files, May 2011. <http://www.cyberengineeringservices.com/malware-obfuscated-within-png-files/>. Last accessed August 12, 2013.
- [30] Cyber Engineering Services. Malware obfuscated within PNG files > sample 2, May 2012. <http://www.cyberengineeringservices.com/malware-obfuscated-within-png-files-sample-2-2/>. Last accessed August 12, 2013.
- [31] Didier Stevens. XORSearch, Jan 2007. <http://blog.didierstevens.com/programs/xorsearch/>. Last accessed August 8, 2013.
- [32] Didier Stevens. Translate, Jul 2008. <http://blog.didierstevens.com/programs/translate/>. Last accessed August 8, 2013.
- [33] Didier Stevens. New tool: XORStrings, Apr 2013. <http://blog.didierstevens.com/?s=xorstrings>. Last accessed August 8, 2013.
- [34] Internet Security Team. Bypassing anti-virus scanners, Mar 2011. <http://dl.packetstormsecurity.net/papers/bypass/bypassing-av.pdf>. Last accessed July 2, 2013.
- [35] Malware Tracker. Cryptam document scanner, 2012. <https://malwaretracker.com/doc.php>. Last accessed July 17, 2013.
- [36] Malware Tracker. New malware document scanner tool released, Feb 2012. <http://blog.malwaretracker.com/2012/02/new-malware-document-scanner-tool.html>. Last accessed August 9, 2013.
- [37] Nikos Virvilis and Dimitris Gritzalis. The big four—what we did wrong in advanced persistent threat detection? In *Proceedings of the 8th International Conference on Availability, Reliability, and Security*, ARES-2013, Sep 2013. <http://www.cis.aueb.gr/Publications/ARES-2013%20APT%20Short.pdf>. Last accessed August 12, 2013.

## **Initial Distribution List**

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Electrical Engineering and Computer Science Department  
West Point  
West Point, New York
4. Science, Math, and Computer Science Program  
Poolesville High School  
17501 W Willard Rd  
Poolesville, MD 20837